Rapport du Projet

Système de Prédiction de l'Efficacité des Employés

24 janvier 2025

Fait par : TAOUSSI Othmane, CHIKHAOUI Ihssane, AIT HMEID LAILA.
Encadré par : AHMED AMAMOU

Contents

1 Introduction

Ce rapport présente le projet de développement d'un système de prédiction de l'efficacité des employés. Ce système utilise des algorithmes d'apprentissage automatique pour analyser les performances des employés et prédire leur efficacité. L'objectif est de fournir aux organisations des outils basés sur les données pour optimiser leur main-d'œuvre et prendre des décisions éclairées.

2 Objectifs du Projet

Les objectifs principaux de ce projet sont :

- Développer un système basé sur l'apprentissage automatique pour prédire l'efficacité des employés.
- Fournir une interface utilisateur intuitive pour saisir les données des employés et afficher les résultats de prédiction.
- Intégrer des fonctionnalités telles que l'analyse en temps réel et des insights basés sur les données.

3 Architecture du Système

Le système est divisé en deux parties principales : le backend et le frontend.

3.1 Backend

Le backend est développé en utilisant Spring Boot et comprend les composants suivants :

- **DemoApplication.java** : Point d'entrée de l'application Spring Boot.
- **EmployeeController.java** : Contrôleur pour gérer les requêtes HTTP et servir les pages web.
- **Employee.java** : Modèle représentant un employé avec ses attributs.
- **PredictionService.java** : Service contenant la logique de prédiction basée sur un modèle de machine learning.

3.2 Frontend

Le frontend est développé en utilisant Thymeleaf et Bootstrap. Les pages principales sont :

- **home.html** : Page d'accueil présentant le système et ses fonctionnalités.
- **index.html** : Formulaire de saisie des données des employés.
- **result.html** : Page affichant les résultats de prédiction.

4 Fonctionnalités

Le système offre les fonctionnalités suivantes :

- **Analyse basée sur l'IA** : Utilisation d'algorithmes de machine learning pour prédire l'efficacité des employés.
- **Insights basés sur les données** : Fournit des insights complets basés sur des indicateurs de performance et des données historiques.
- **Prédictions en temps réel** : Permet de faire des prédictions instantanées pour une prise de décision rapide.

5 Technologies Utilisées

- **Spring Boot** : Framework Java pour le développement d'applications web.
- **Thymeleaf** : Moteur de template pour les pages web.
- **Bootstrap** : Framework CSS pour le design responsive.
- **Deeplearning4j** : Bibliothèque Java pour le machine learning.

6 Interface Utilisateur

Voici les captures d'écran des différentes pages de l'application.

6.1 Page d'Accueil

La page d'accueil présente une introduction au système et ses fonctionnalités principales.

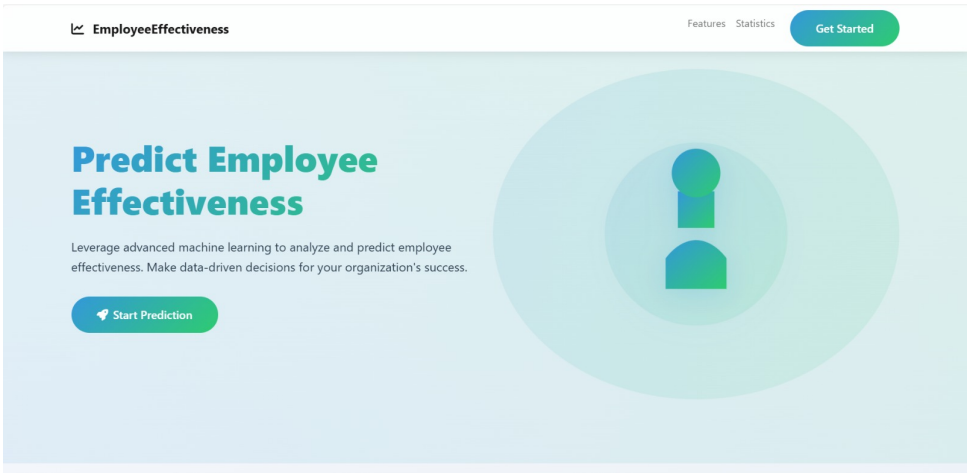

Figure 1: Page d'accueil du système.

6.2 Formulaire de Prédiction

Le formulaire permet à l'utilisateur de saisir les informations de l'employé pour effectuer une prédiction.

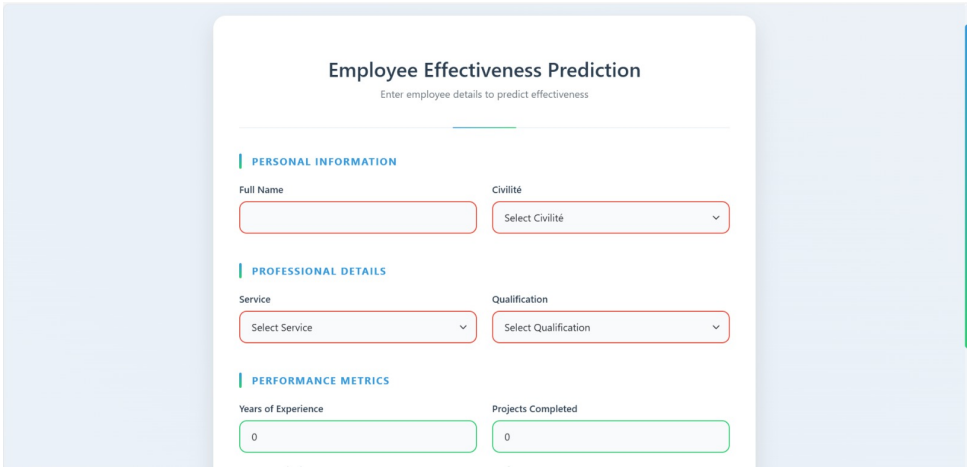

Figure 2: Formulaire de saisie des données de l'employé.

6.3 Résultats de Prédiction

La page des résultats affiche l'efficacité prédite de l'employé ainsi que les détails saisis.

6.4 Statistiques et Fonctionnalités

Les captures suivantes montrent les statistiques et les fonctionnalités clés du système.

7 Exemples de Code

Voici quelques extraits de code clés du projet.

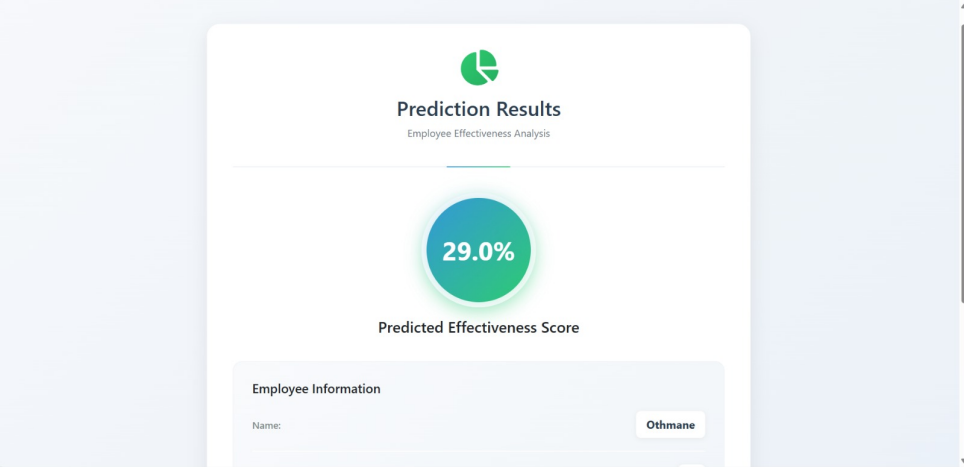

Figure 3: Résultats de la prédiction.

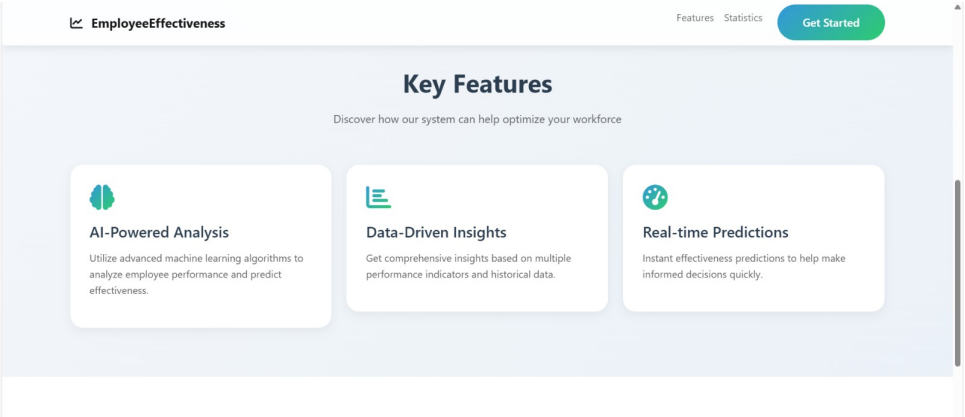

Figure 4: Statistiques et fonctionnalités du système.

7.1 EmployeeController.java

```java
package com.example.demo.controller;

import com.example.demo.model.Employee;
import com.example.demo.service.PredictionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class EmployeeController {

    @Autowired
    private PredictionService predictionService;

    @GetMapping("/")
    public String home() {
        return "home";
    }

    @GetMapping("/predict")
    public String showForm(Model model) {
        model.addAttribute("employee", new Employee());
        return "index";
    }
```

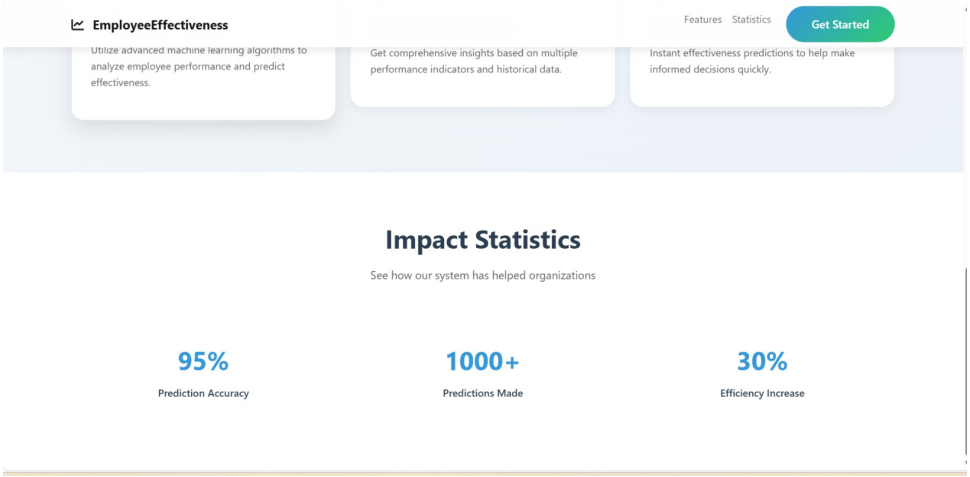

Figure 5: Précision des prédictions et efficacité.

```
28
29      @PostMapping("/predict")
30      public String predict(@ModelAttribute Employee employee, Model model) {
31          double effectiveness = predictionService.predictEffectiveness(employee);
32          employee.setPredictedEffectiveness(effectiveness);
33          model.addAttribute("employee", employee);
34          return "result";
35      }
36  }
```

7.2 PredictionService.java

```
1   package com.example.demo.service;
2
3   import com.example.demo.model.Employee;
4   import org.deeplearning4j.nn.modelimport.keras.KerasModelImport;
5   import org.deeplearning4j.nn.multilayer.MultiLayerNetwork;
6   import org.nd4j.linalg.api.ndarray.INDArray;
7   import org.nd4j.linalg.factory.Nd4j;
8   import org.springframework.stereotype.Service;
9   import org.springframework.util.ResourceUtils;
10
11  import javax.annotation.PostConstruct;
12  import java.io.File;
13  import java.util.Map;
14  import java.util.HashMap;
15
16  @Service
17  public class PredictionService {
18
19      private MultiLayerNetwork model;
20      private final Map<String, Integer> civiliteMap;
21      private final Map<String, Integer> sitFamMap;
22      private final Map<String, Integer> situationMap;
23      private final Map<String, Integer> serviceMap;
24      private final Map<String, Integer> qualifMap;
25
26      public PredictionService() {
27          // Initialisation des mappings pour les variables cat gorielles
28          civiliteMap = new HashMap<>();
29          civiliteMap.put("MLE", 0);
30          civiliteMap.put("MR", 2);
31          civiliteMap.put("MME", 1);
32
33          sitFamMap = new HashMap<>();
34          sitFamMap.put("couple", 0);
35          sitFamMap.put("marriee", 2);
36          sitFamMap.put("divorce", 1);
```

```
37
38          situationMap = new HashMap<>();
39          situationMap.put("horraire", 0);
40          situationMap.put("mensuel", 1);
41
42          serviceMap = new HashMap<>();
43          serviceMap.put("direction", 1);
44          serviceMap.put("RH ET finance", 2);
45          serviceMap.put("logistiques", 3);
46          serviceMap.put("lsm", 4);
47          serviceMap.put("sogec coupons", 5);
48          serviceMap.put("facility", 6);
49          serviceMap.put("tessi", 7);
50          serviceMap.put("sogec-odr", 8);
51          serviceMap.put("informatique", 9);
52          serviceMap.put("smq", 10);
53          serviceMap.put("sogec masque", 11);
54          serviceMap.put("optimal", 12);
55          serviceMap.put("central contact", 13);
56          serviceMap.put("brandbank", 14);
57          serviceMap.put("NEANT", 15);
58          serviceMap.put("TAXIM", 16);
59          serviceMap.put("YOCAST", 17);
60
61          qualifMap = new HashMap<>();
62          qualifMap.put("grant", 1);
63          qualifMap.put("tech de maintenace", 2);
64          qualifMap.put("responsable stock", 3);
65          qualifMap.put("operateur", 5);
66          qualifMap.put("agent de qualite", 6);
67          qualifMap.put("assistantant resp projet", 7);
68          qualifMap.put("superviseur moduction", 8);
69          qualifMap.put("assistant resp informatique", 9);
70          qualifMap.put("agent logistique", 10);
71          qualifMap.put("directeur", 11);
72          qualifMap.put("responsable comptabilite", 12);
73          qualifMap.put("agent de soutien", 13);
74          qualifMap.put("directeur de production", 14);
75          qualifMap.put("RH", 15);
76          qualifMap.put("responsable systeme and rh", 16);
77          qualifMap.put("agent manutention", 17);
78          qualifMap.put("assistant superviseur de production", 18);
79          qualifMap.put("assistant administrative", 19);
80          qualifMap.put("Responsable achat et logistique", 20);
81          qualifMap.put("operateur anapec", 21);
82          qualifMap.put("teleconseiller", 22);
83          qualifMap.put("tele operateur", 23);
84          qualifMap.put("agent de production", 24);
85          qualifMap.put("responsable plateau", 25);
86          qualifMap.put("developeur informatique", 26);
87          qualifMap.put("resp developeur informatique", 27);
88          qualifMap.put("directeur de si", 28);
89          qualifMap.put("community manager", 29);
90          qualifMap.put("infographiste", 30);
91      }
92
93      @PostConstruct
94      public void init() {
95          try {
96              String modelPath = "C:\\Users\\taous\\Desktop\\Deploiement_Model\\model.h5";
97              File modelFile = new File(modelPath);
98
99              if (!modelFile.exists()) {
100                 throw new RuntimeException("Model file not found at: " + modelPath);
101             }
102
103             // Chargement du mod le
104             model = KerasModelImport.importKerasSequentialModelAndWeights(modelPath);
105             System.out.println("Model loaded successfully from: " + modelPath);
106
107         } catch (Exception e) {
108             System.err.println("Failed to load model: " + e.getMessage());
109             e.printStackTrace();
```

```
        }
    }

    public double predictEffectiveness(Employee employee) {
        try {
            if (model == null) {
                throw new RuntimeException("Model not initialized");
            }

            // Conversion des variables cat gorielles en valeurs num riques
            double[] features = {
                civiliteMap.get(employee.getCivilite_salarie()),
                qualifMap.get(employee.getQualif_salarie()),
                situationMap.get(employee.getSituation_salarie()),
                serviceMap.get(employee.getService_salarie()),
                sitFamMap.get(employee.getSit_fam_salarie()),
                employee.getYearsExperience(),
                employee.getProjectsCompleted(),
                employee.getHoursWorked(),
                employee.getPerformanceScore()
            };

            // Pr diction
            INDArray input = Nd4j.create(features).reshape(1, features.length);
            INDArray output = model.output(input);
            double predictedValue = output.getDouble(0);

            // Mise    l' chelle  de la pr diction entre 0 et 100
            return Math.min(Math.max(predictedValue * 100, 0), 100);

        } catch (Exception e) {
            System.err.println("Prediction error: " + e.getMessage());
            e.printStackTrace();

            // Retourner une valeur par d faut en cas d'erreur
            return (employee.getYearsExperience() * 0.25
                    + employee.getProjectsCompleted() * 0.25
                    + employee.getHoursWorked() * 0.25
                    + employee.getPerformanceScore() * 0.25);
        }
    }
}
```

8 Conclusion

Le système de prédiction de l'efficacité des employés est un outil puissant pour les organisations cherchant à optimiser leur main-d'œuvre. En combinant des algorithmes de machine learning avec une interface utilisateur intuitive, ce système offre des insights précieux pour une prise de décision éclairée. Les futures améliorations pourraient inclure l'intégration de modèles plus avancés et l'ajout de fonctionnalités supplémentaires pour une analyse plus approfondie.